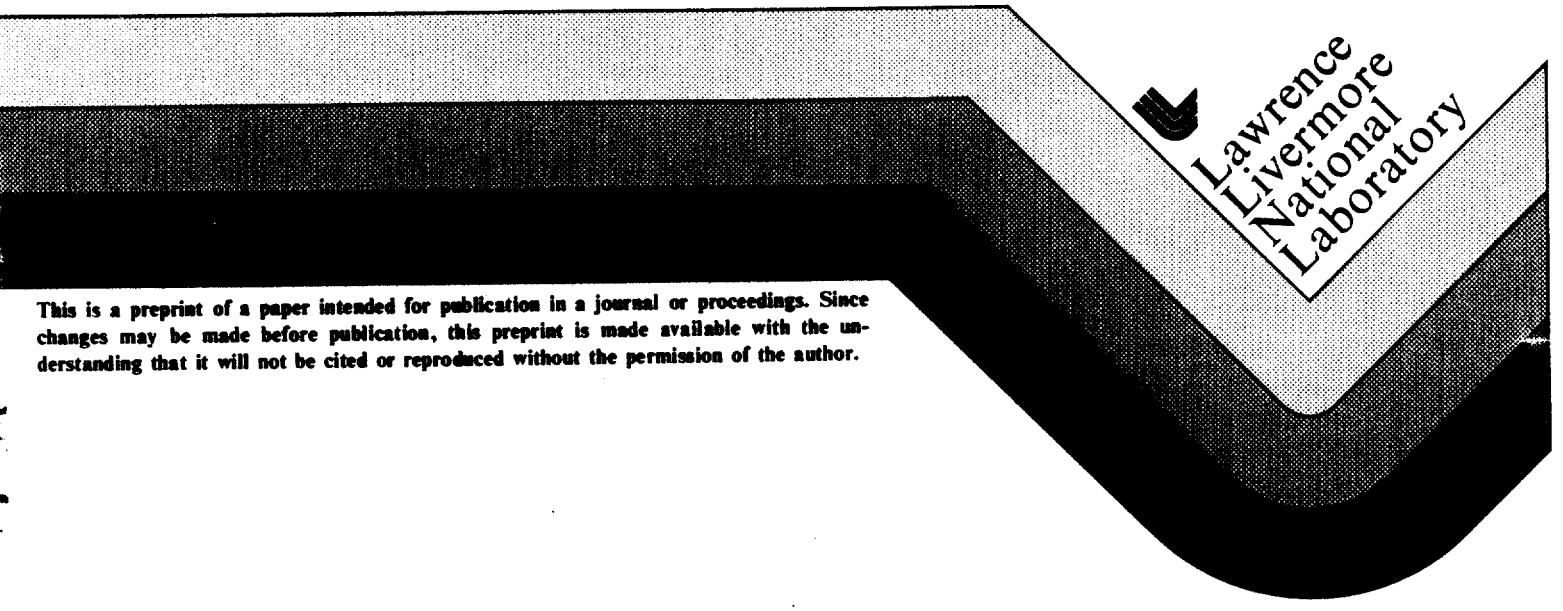
UCRL-91032
PREPRINT

# Benchmarking-GraphicsWorkstations

Bruce Eric Brown
Robert L. Judd

This paper was prepared for submittal to
IEEE Conference on Computer Workstations
Red Lyon Inn, San Jose, CA
November 12-14, 1985

September 11, 1985



Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-ENG-48.

# BENCHMARKING-GraphicsWorkstations

Bruce Eric Brown and Robert L. Judd*

The Quail Group, Orem, Utah and University of California, Lawrence Livermore National Laboratory, Livermore, California

*This paper reports on work being performed to benchmark a new breed of machines, the graphics workstations for scientific and engineering applications. We begin with the history of workstations and how they developed. Many different forces created the graphics workstation. Scientific and engineering calculations are traditionally performed by super computers in a computing center. The graphic workstation was introduced, and this technology was added to the computing environment. The answers we need are...what can the workstations do and when do we remove the programs or parts of programs from supercomputers and place them on workstations? Benchmarks and what they can and can not measure are discussed in this light. The particulars about graphics benchmarks and the operating system, including windows, are reviewed, then the benchmarks we are using are described and the results are presented. Since this work is never finished, future plans are also discussed. The emphasis of our work is toward scientific and engineering workstations connected with existing mainframe computers.*

*In the industry today we hear the term JAWS (Just Another WorkStation) when a new workstation is introduced. What does the new product do beyond those on the market? In this paper we present ways to measure and tell the differences between workstations.*

## 1. Introduction

What is a graphics workstation? Is it a PC, a terminal, a mini-computer, or a micro? It is these and more. In the beginning, in about the mid 1960's, we had the IBM 360 and the CDC 6600 with operating systems allowing for batch and timesharing on a large scale. The interactive terminal was the old Teletype model 33. Graphics systems were available, but they were large and expensive. By the late 1960's we had introduced the direct views storage tube (DVST), otherwise known as the Tektronix. Graphics became available to more users. The computer networks were developing and smaller graphics system (calligraphic) were used. Video images generated on the computer could be shown and refreshed from disks, and in the operating system world UNIX began.

* Authors' current addresses: Bruce Eric Brown, The Quail Group, Inc., 561 East Quail Road, Orem, Utah 84057 (801)225-4342 and Robert L. Judd, Los Alamos National Laboratory, C-6 Computer Graphics, MS-B272, P.O. Box 1663, Los Alamos, New Mexico 87545 (505)667-7356.

In the early 1970's Texas Instruments introduced the Silent 700 terminal and "dumb" video or glass terminals were used. The mini-computer developed a large following requiring more terminals and more local capabilities. Toward the late 1970's the XEROX PARC (Palo Alto Research Center) used the ALTO computer, windows, and Ethernet. This was a move toward graphics workstations. The late 1970's also brought "smart" terminals, color graphics (bit-mapped), "home" computers, microprocessors, CP/M, and add-on graphics processors. Terminals got smarter and computers went home. In the 1980's, "intelligent" terminals arrived, along with local area networks, the first graphics workstations, and the IBM PC with MSDOS. UNIX was also supported by AT&T.

The result was that graphics workstations grew out of terminals, mini-computers, PCs, data communications, networks, and research and development advances. They are a combination of many developments and they are still evolving (see Figure 1). We can not give a definitive definition of a graphics workstation, but some of its major attributes can be described. A graphics workstation is micro-computer based with a disk operating system. It has a graphics display and a data communication interface. That is the minimum. More advanced features of graphics workstations include bit mapped displays, higher resolution (1024+), window systems with mice, local area networks, and multi-process and multi-user operating systems. The computing power also exceeds many of the 1960's mainframes.

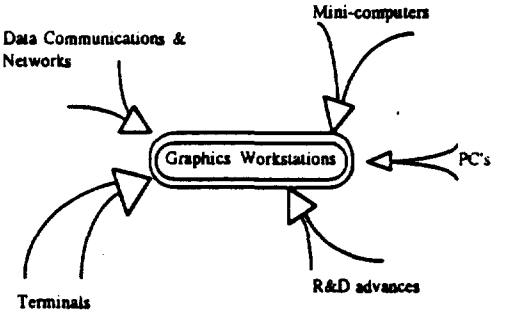



Figure 1. Influences on Workstations

Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

## 2. Benchmarking

Why Benchmark? The answer is obvious. We need to compare systems before we purchase them. Who has not been burned with "caveat emptor"? Do we believe the manufacturer's claims or do we do our own testing? Answers are needed before we buy. Today we need to know what graphics workstations can do and how we can distribute our computing needs over them in conjunction with our existing mainframes. To increase productivity, some programs should be taken off the mainframe systems and placed on the workstations-but which ones and when are the major questions. Benchmarks can help us in our planning, but what is a benchmark?

From the dictionary we read that a benchmark is a marked point of known or assumed elevation. In Byte magazine (Feb. 1984) Jerry Houston said: "The ideal benchmark...a carefully planned demonstration in which the specific application intended for the product is simulated as closely as possible."(1) Can each potential buyer take the time to develop benchmarks before each purchase? The answer is no. We do not have the luxury of time or resources to do this. Potential purchasers want an EPA highway and city estimate of the MPG for the systems they are considering. A benchmark should be a simulation of use. Short of leasing the equipment for an extended period of time, we can not test the use before we buy.

For graphics workstations, we are concerned with: graphics, processing power (both floating point and integer), and the operating system (disks, networks, languages and windows). The functionality and performance need to be measured. The benchmarks do not consider the ergonomics, heat, noise, power requirements or looks. The usefulness or useability that are critical to the productivity of the system are also not considered.

Benchmarks are available and much has been written about them (2,3). We have the Whetstones, the tower of Hanoi, and a set of commercially available benchmarks for UNIX from AIM technology. For the processing power, particularly floating point there are several known benchmarks. The Livermore Fortran Kernels (4) and the Argonne LINPACK results (5) to name just two. These measure the speed and report the values in FLOPS (floating point operations per second). The emphasis of floating point benchmarks has been to measure the speed. Karpinski (6) has also published an article on the quality or correctness of the floating point calculations. Another measure of the speed of the system are the numbers reported by John Swanson of SASI for his ANSYS finite element program. This is not a benchmark, but rather the result of a large number of problems run on each different machine(7). In the next section we look at what is needed for a graphics benchmark.

## 3. Graphics

When we discuss graphics, each application has different needs. Are the plots two-dimensional or three-dimensional? Do we need color? How fast do we need to interact with the images produced? To address these issues we need to measure several different functions of the workstation. The first is the raw drawing speed of the system. When we use calligraphic systems the vector inches per second are usually readily available and accurate. For raster systems the vector inches per second are a little harder to obtain and are not that meaningful. We need to measure the scan conversion time of the workstations in terms of pixels per second but this has to be used carefully in our evaluation of systems, since a vector of the same screen length on a 512 x 512 pixel system has one half as many pixel calculations as on a 1024 x 1024 system. To realize the same screen display rate, the 1024 x 1024 system must draw vectors twice as fast as the 512 x 512 system.

The second category to be measured is transformation times, including clipping, rotations (if any), and scaling. The two-dimensional rotations may or may not be present in the system. At this level we also have several packages of callable routines. The native mode of the hardware usually has a hook here and then libraries such as GKS and PLOT10 are also available.

Much of the work of modern scientific and engineering calculations uses three-dimensional data--hence the need for three-dimensional displays. This adds to the amount of processing required to display a picture. The data can then be rotated, scaled, clipped, and projected onto a two-dimensional screen before being displayed. Libraries of routines for these functions such as ACM SIGGRAPH GSPC Core or ISSCO's DISPPLA are available. Hidden surface removal or hidden line removal must also be taken care of, but that is currently outside the scope of this work. The transformations of three-dimensional data take floating point processing power. Some workstations have special VLSI processors for these functions (8), while others use general-purpose hardware. The benchmarks try to measure the different modes of plotting.

## 4. O/S and Windows

When we look at workstations, we have complete computer systems in their own right. The choices seem to be two: the first is an open system to which any other manufacturer of products can run, and the second is a closed system. A good example of the open system is the SUN Computer system running UNIX 4.2. Any manufacturer who uses this operating system can be added to the network. A good example of the closed system is Apollo.

Windows have become a requirement for some users. Most serious users have more than one problem or task running at the same time. They may have a calculation running while they prepare data for the next run and work on a progress report due that day. Windows allow the user to do more than one task or at least monitor more than one task at the same time. Interaction with the systems is usually done with a mouse although tablets, joy sticks, and touch systems also work.

## 5. The Benchmarks

The benchmarks used for evaluation of graphics workstations include much of the work of previous benchmarks. We have started with the Livermore Fortran Kernels for floating point performance and have added the graphics benchmarks. The first graphics benchmark attempts to determine the raw drawing speed of the graphics device. This is done by sending vectors of known length to the device at the lowest level to be plotted. To accomplish this we have had to develop our own library of graphics functions. This library consists of open, close, erase, line, move, draw, and timing functions. Appendix A gives a listing of one of these libraries for the SUN computer. The second benchmark

uses the Sierpinski space filling curve algorithm (9) to plot two-dimensional vectors. The third benchmark attempts to determine the time that the processor would take to do the transformation required to plot three-dimensional data. The last benchmark plots a "standard" picture in both two and three dimensions.

The first graphics benchmark tries to calculate the speed of the scan conversion for vectors in two dimensions. Three loops are used. the first draws 1000 (default) long vectors. A long vector is defined to be the minimum of the screen size in X and Y. Next it draws the same number of short vectors. Short here is defined to be 1/20th of a long vector. The final loop calls a dummy routine the same number of times so we can subtract the system overhead of the function calls. The vectors are also plotted at various angles, from 0 to 90 degrees at 5 degree increments (see Figure 2). This is to determine if there are any angular dependencies for the scan conversion algorithm. Looking at the results presented below we can also see the influence of the hardware architecture: horizontal vectors can be plotted faster than non-horizontal vectors.

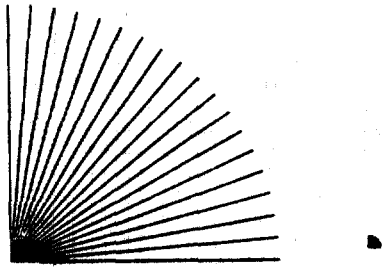


Figure 2. First Benchmark output.

The results of the benchmark are printed out and shown below.

| Angle | Long=1024, | Short=51, | Dummy | VPS L | VPS S | Pixels/sec |
|---|---|---|---|---|---|---|
| 0 | 6.050 | 5.617 | 0.050 | 165.2 | 178.0 | 2245384.6 |
| 5 | 6.633 | 6.050 | 0.033 | 150.7 | 165.3 | 1660000.0 |
| 10 | 7.033 | 6.100 | 0.033 | 142.1 | 163.9 | 1042500.0 |
| 15 | 7.100 | 6.083 | 0.050 | 140.8 | 164.3 | 957049.2 |
| 20 | 7.700 | 6.083 | 0.033 | 129.8 | 164.4 | 601855.7 |
| 25 | 8.133 | 6.117 | 0.050 | 122.9 | 163.4 | 482479.3 |
| 30 | 8.033 | 6.133 | 0.033 | 124.4 | 163.0 | 512105.3 |
| 35 | 8.833 | 6.150 | 0.033 | 113.2 | 162.6 | 362608.7 |
| 40 | 8.783 | 6.167 | 0.050 | 113.8 | 162.1 | 371847.1 |
| 45 | 8.733 | 6.183 | 0.017 | 114.5 | 161.7 | 381568.6 |
| 50 | 8.567 | 6.167 | 0.017 | 116.7 | 162.1 | 405416.7 |
| 55 | 8.767 | 6.150 | 0.017 | 114.1 | 162.6 | 371847.1 |
| 60 | 8.700 | 6.217 | 0.017 | 114.9 | 160.8 | 391812.1 |
| 65 | 8.617 | 6.150 | 0.033 | 116.0 | 162.6 | 394459.5 |
| 70 | 8.300 | 6.150 | 0.017 | 120.5 | 162.6 | 452558.1 |
| 75 | 8.233 | 6.150 | 0.017 | 121.4 | 162.6 | 467040.0 |
| 80 | 7.950 | 6.133 | 0.017 | 125.8 | 163.0 | 535596.3 |
| 85 | 7.767 | 6.167 | 0.017 | [illegible] | 162.1 | 608125.0 |
| 90 | 7.067 | 5.700 | 0.033 | [illegible] | 175.4 | 711951.2 |
| | | Average | | 12[illegible] | 164.3 | 682326.6 |

The results above are for the Sun workstation and we interpret the numbers as the horizontal data goes faster by a factor of six over the other angles. When lines are close to horizontal they also go faster because of the hardware layout. The pixel speed of the hardware after all of the overhead is eliminated is the average pixels per second report, almost 700,000 pixels per second. The ratio of the short to long vector plot times is nowhere near the twenty-to-one ratio of their lengths so the overhead of setup for a vector is significant.

The second benchmark is named Sierpinski. This program generates a curve from a family of space-filling curves (see Figure 3). The algorithm taken to its limits will fill a two-dimensional plane. This benchmark was chosen because it generates a large number of vectors and they become progressively shorter. Many engineering and scientific plots as the complexity increases the vector length becomes shorter so this is an approximation of these types of plots. The times for performing these algorithms are also shown in table form.

```
SIERP -- GRAPHICS OFF
order  1 -- sec =  0.02 -- vec =     16 -- vec/sec =    960
order  2 -- sec =  0.02 -- vec =     80 -- vec/sec =   4800
order  3 -- sec =  0.02 -- vec =    336 -- vec/sec =  20160
order  4 -- sec =  0.08 -- vec =   1360 -- vec/sec =  16320
order  5 -- sec =  0.35 -- vec =   5456 -- vec/sec =  15588
order  6 -- sec =  1.40 -- vec =  21840 -- vec/sec =  15600
order  7 -- sec =  5.63 -- vec =  87376 -- vec/sec =  15510
order  8 -- sec = 22.65 -- vec = 349520 -- vec/sec =  15431

SIERP -- GRAPHICS ON
order  1 -- sec =  0.12 -- vec =     16 -- vec/sec =    137
order  2 -- sec =  0.48 -- vec =     80 -- vec/sec =    165
order  3 -- sec =  1.97 -- vec =    336 -- vec/sec =    170
order  4 -- sec =  7.82 -- vec =   1360 -- vec/sec =    173
order  5 -- sec = 31.03 -- vec =   5456 -- vec/sec =    175
```

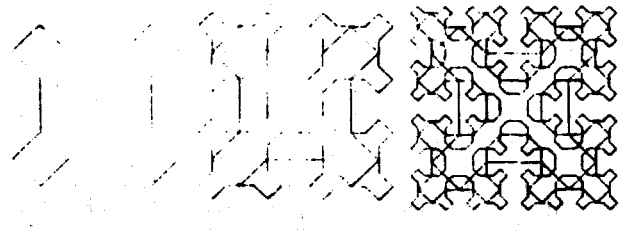


Figure 3. Sierpinski, order 1, 2, and 3.

The nature of the program is to shut itself off after the first curve of order n takes more than twenty seconds. The program also draws shorter vectors as the order increases. The numbers generated here in vectors per second average 164, which matches the average for short vectors reported by our first benchmark.

The third graphics benchmark is related to the Livermore Fortran Kernels. In this we do the calculations necessary for transforming data from three-dimensional World coordinates (WC) to Device Coordinates (DC) and include clipping and perspective. To accomplish all of this the transformation of rotation about each axis (X,Y, and Z), translation in X,Y, and Z, scaling of the data, and the preparation for perspective viewing are performed with a 4x4 matrix multiplication (8). The clipping is done with IF tests and then the perspective X,Y coordinates are determined with the division by the scaled Z coordinate. The algorithm is shown below written in C:

```
for( i=0; i<num_vectors; i++)
  {
/* transformation, rotate, translate and scale */

   x=vecs[i][0]*m[0][0] + vecs[i][1]*m[1][0] + vecs[i][2]*m[2][0] +m[3][0];
   y=vecs[i][0]*m[0][1] + vecs[i][1]*m[1][1] + vecs[i][2]*m[2][1] +m[3][1];
   z=vecs[i][0]*m[0][2] + vecs[i][1]*m[1][2] + vecs[i][2]*m[2][2] +m[3][2];
   w = vecs[i][2] + m[3][3];

/* clip */

   if( x > w ) x=w;
   else if( x < -w ) x = -w;

   if( y > w ) y=w;
   else if( y < -w ) y = -w;

   if( z > w ) z=w;
   else if( z < -w ) z = -w;

/* perspective division */

   x = x/z;
   y = y/z;
   }
```

Results are again printed out and shown below for the SUN2/100 without the floating point processor. The test was run ten times to obtain the average.

```
time =     4.740   210.970 trans. per sec. =     6.962 KFLOPS
time =     4.540   220.264 trans. per sec. =     7.269 KFLOPS
time =     4.600   217.391 trans. per sec. =     7.174 KFLOPS
time =     4.340   230.415 trans. per sec. =     7.604 KFLOPS
time =     4.200   238.095 trans. per sec.       7.857 KFLOPS
time =     4.380   228.311 trans. per sec. =     7.534 KFLOPS
time =     4.900   204.082 trans. per sec. =     6.735 KFLOPS
time =     4.420   226.244 trans. per sec. =     7.466 KFLOPS
time =     4.200   238.095 trans. per sec. =     7.857 KFLOPS
time =     4.240   235.849 trans. per sec. =     7.783 KFLOPS
     Averages      224     trans. per sec.       7.46  KFLOPS
```

The space shuttle plot shown in Figure 4 is our standard three-dimensional plot. We first take the three-dimensional data and transform it to two dimensions and plot it. The two-dimensional display list is then plotted and finally a dummy routine is called to plot the data. These times are reported in table form:

```
 1104 vectors having a total length of 38316 (pixels).
Average vector length = 34, max = 361, min = 1
3D time = 15.050 secs,   73.4 vecs/sec or  2545.9 pixels/sec
2D time =  6.300 secs,  175.2 vecs/sec or  6081.9 pixels/sec
2Dltime =  0.500 secs, 2208.0 vecs/sec or 54046.0 pixels/sec
```

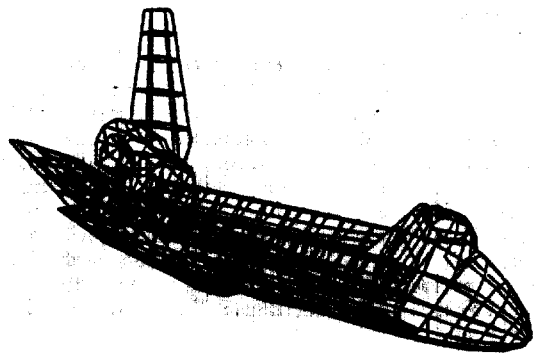


Figure 4. Space Shuttle - Standard Picture

To check our data we take the times reported by our first benchmark for plotting and the third for transforming, and we find that the 1104 vectors would take approximately 9+ seconds to transform and 5+ seconds to plot. We see that the two-dimensional plot time is 6.3 seconds, so we are within a reasonable error range.

The results presented are a function of being able to measure accurately the time it takes to perform each operation. Most systems have timing functions available and our requirements are that the benchmarks be run long enough that the errors in timing are small. A great statement, but what does it mean? We run each benchmark several times and try to take the best numbers produced. On UNIX systems there are many daemons running around in the background which can interfere with our runs if the timing routines do not take them into account. Below are shown two sets of numbers generated on the SUN one after another. Attempts were made to delete every possible conflicting daemon, but yet the numbers differ. Fortunately, the averages are pretty good. The accurate timing of runs should be the subject of another paper.

Loop Number 1 of 2 Number of Vectors = 1000 Length = 1024

| Angle | Long=1024, | Short=51, | Dummy | VPS L | VPS S | Pixels/sec |
|---|---|---|---|---|---|---|
| 0 | 6.717 | 5.617 | 0.017 | 148.9 | 178.0 | 884545.5 |
| 5 | 6.633 | 6.033 | 0.017 | 150.7 | 165.7 | 1621666.7 |
| 10 | 7.050 | 6.13 | 0.033 | 141.8 | 163.0 | 1061454.5 |
| 15 | 7.067 | 6.067 | 0.017 | 141.5 | 164.8 | 973000.0 |
| 20 | 7.683 | 6.133 | 0.017 | 130.1 | 163.0 | 627741.9 |
| 25 | 8.133 | 6.133 | 0.017 | 122.9 | 163.0 | 486500.0 |
| 30 | 8.117 | 6.117 | 0.017 | 123.2 | 163.5 | 486500.0 |
| 35 | 8.817 | 6.167 | 0.033 | 113.4 | 162.1 | 367169.8 |
| 40 | 8.767 | 6.167 | 0.017 | 114.1 | 162.1 | 374230.8 |
| 45 | 8.717 | 6.150 | 0.033 | 114.7 | 162.6 | 379090.9 |
| 50 | 8.600 | 6.167 | 0.017 | 116.3 | 162.1 | 399863.0 |
| 55 | 8.750 | 6.183 | 0.033 | 114.3 | 161.7 | 379090.9 |
| 60 | 8.717 | 6.200 | 0.050 | 114.7 | 161.2 | 386622.5 |
| 65 | 8.650 | 6.133 | 0.033 | 115.6 | 163.0 | 386622.5 |
| 70 | 8.300 | 6.150 | 0.050 | 120.4 | 162.6 | 452558.1 |
| 75 | 8.217 | 6.117 | 0.033 | 121.7 | 163.5 | 463333.3 |
| 80 | 7.950 | 6.150 | 0.033 | 125.8 | 162.6 | 540555.6 |
| 85 | 7.750 | 6.150 | 0.050 | 129.0 | 162.6 | 608125.0 |
| 90 | 7.050 | 5.700 | 0.033 | 141.8 | 175.4 | 720740.7 |
| | | Average | | 126.4 | 164.3 | 610495.4 |

Loop Number 2 of 2 Number of Vectors = 1000 Length = 1024

| Angle | Long=1024, | Short=51, | Dummy | VPS L | VPS S | Pixels/sec |
|---|---|---|---|---|---|---|
| 0 | 6.050 | 5.617 | 0.050 | 165.2 | 178.0 | 2245384.6 |
| 5 | 6.633 | 6.050 | 0.033 | 150.7 | 165.3 | 1668000.0 |
| 10 | 7.033 | 6.100 | 0.033 | 142.1 | 163.9 | 1042500.0 |
| 15 | 7.100 | 6.083 | 0.050 | 140.8 | 164.3 | 957049.2 |
| 20 | 7.700 | 6.083 | 0.033 | 129.8 | 164.4 | 601855.7 |
| 25 | 8.133 | 6.117 | 0.050 | 122.9 | 163.4 | 482479.3 |
| 30 | 8.033 | 6.133 | 0.033 | 124.4 | 163.0 | 512105.3 |
| 35 | 8.833 | 6.150 | 0.033 | 113.2 | 162.6 | 362608.7 |
| 40 | 8.783 | 6.167 | 0.050 | 113.8 | 162.1 | 371847.1 |
| 45 | 8.733 | 6.183 | 0.017 | 114.5 | 161.7 | 381568.6 |
| 50 | 8.567 | 6.167 | 0.017 | 116.7 | 162.1 | 405416.7 |
| 55 | 8.767 | 6.150 | 0.017 | 114.1 | 162.6 | 371847.1 |
| 60 | 8.700 | 6.217 | 0.017 | 114.9 | 160.8 | 391812.1 |
| 65 | 8.617 | 6.150 | 0.033 | 116.0 | 162.6 | 394459.5 |
| 70 | 8.300 | 6.150 | 0.017 | 120.5 | 162.6 | 452558.1 |
| 75 | 8.233 | 6.150 | 0.017 | 121.4 | 162.6 | 467040.0 |
| 80 | 7.950 | 6.133 | 0.017 | 125.8 | 163.0 | 535596.3 |
| 85 | 7.767 | 6.167 | 0.017 | 128.7 | 162.1 | 608125.0 |
| 90 | 7.067 | 5.700 | 0.033 | 141.5 | 175.4 | 711951.2 |
| | | Average | | 127.2 | 164.3 | 682326.6 |

Results are also given for the SUN system using the windows. We have the further luxury of being able to lock the screen for any other updates while we plot our

data. This significantly improves the performance. The Sierpinski algorithm was also run using the different plotting packages available. The results are shown below. These sample executions were run on a SUN2/100 under their window system.

```
Sierpinski with the standard SUNCORE plotting package.

order  1 - sec =  0.12 - vec =     16 -- vec/sec =   137
order  2 - sec =  0.48 - vec =     80 -- vec/sec =   165
order  3 - sec =  1.97 - vec =    336 -- vec/sec =   170
order  4 - sec =  7.82 - vec =   1360 -- vec/sec =   173
order  5 - sec = 31.03 - vec =   5456 -- vec/sec =   175

Sierpinski with the SUN CGI plotting package.

order  1 - sec =  0.08 - vec =     16 -- vec/sec =   192
order  2 - sec =  0.37 - vec =     80 -- vec/sec =   218
order  3 - sec =  1.55 - vec =    336 -- vec/sec =   216
order  4 - sec =  6.02 - vec =   1360 -- vec/sec =   226
order  5 - sec = 23.60 - vec =   5456 -- vec/sec =   231

Sierpinski with the fast absolute plotting package with
SUN Windows

order  1 - sec =  0.02 - vec =     16 -- vec/sec =   960
order  2 - sec =  0.03 - vec =     80 -- vec/sec =  2400
order  3 - sec =  0.17 - vec =    336 -- vec/sec =  2016
order  4 - sec =  0.58 - vec =   1360 -- vec/sec =  2331
order  5 - sec =  2.05 - vec =   5456 -- vec/sec =  2661
order  6 - sec =  7.23 - vec =  21840 -- vec/sec =  3019
order  7 - sec = 25.53 - vec =  87376 -- vec/sec =  3422
```

## 6. Summary of Results

The results of these benchmarks plus the LLNL Fortran Kernels are reported in Appendix B. The tables first summarize the systems tested by detailing the hardware processor, the clock speed, the presence of floating point accelerators and vector processors, the size of memory, disk space, graphics device, operating systems and local area network. The second table lists the results of the LLNL Kernels and LINPACK when available along with our benchmark number three. The next table presents the results of our first benchmark in three different modes. The first is the lowest level graphics, the second is the standard or library graphics and the third set is the windowed graphics. The numbers reported are the short and long vector pixels per second.

The Sierpinski curve was used to test these systems since its vectors are progressively shorter and it generates a large number of them. The plotting speeds are again reported in vectors per second for the three cases of graphics routine described above. The last table details the space shuttle, our standard plot, again with the three sets of graphics numbers.

## 7. Future Work

This is a report of the methodology being used to benchmark workstations. The emphasis of this paper has been on the work for the graphics portion of the benchmarking. The effort at LLNL is to the whole picture of benchmarks. We are continuing to refine the routines and to test various vendors hardware.

We have developed four benchmarks that relate well to vector plots required by engineering and scientific users. As the suite of programs evolves we will include the space shuttle as a standard plot for color and continuous-tone images using polygons. We also expect to test and rate hidden surface removal algorithms and hardware. The work of benchmarking will never be finished.

## 8. References


(1) Houston, Jerry, "Don't Bench Me In," Byte, Volume 9, Number 2, February 1984, pp. 160-167.

(2) Bemwell, N., Benchmarking, J. Wiley and Sons, Toronto, 1975.

(3) Spooner, C.R., "Benchmarking Interactive Systems: Modeling the Application," Proceedings of the 15th Meeting of the Computer Performance Evaluation Users Group (CPEUG), pp. 53-63.

(4) McMahon, Frank H., "The Livermore Fortran Kernels: A CPU Floating Point Performance Test," Lawrence Livermore National Laboratory, Livermore, CA (to be published).

(5) Dongarra, Jack J., "Performance of Various Computers Using Standard Linear Equations Software in a Fortran Environment," Argonne National Laboratory, Mathematics and Computer Science Division, Technical Memorandum No. 23, December, 1984.

(6) Karpinski, Richard, "Paranoia: A Floating-Point Benchmark," Byte, Volume 10, Number 2, February 1985, pp. 223-235.

(7) Private communication, January 1985.

(8) Clark, James, "The Geometry Engine: A VLSI Geometry System for Graphics," Computer Graphics, Volume 16, Number 3, pp.127-133.

(9) Wirth, N., ALGORITHMS+DATA STRUCTURES = PROGRAMS, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1976, pp. 136-137.

## Appendix A - Listing of Graphic Functions Library

```
/*=======================================
/*
/*  Standard Core graphics library for SunCore
/*
/*  This work was produced under the sponsorship of the
/*  the U.S. Department of Energy
/*
/*  (c) copyright 1985 by the Regents of the University
/*      of California.  All rights reserved
/*
/*   programmed by Bob Judd for LLNL benchmark work
/*  version 1a: 15 May 1985
/*=======================================
#include <sys/types.h>
#include <sys/times.h>
#include <stdio.h>
#include <usercore.h>
char suffix[2] = "b";
int bwldd();
int pixwindd();
struct vwsurf rawsurface    = DEFAULT_VWSURF(bwldd);
struct vwsurf windowsurface = DEFAULT_VWSURF(pixwindd);
struct pixwin *pw;
struct rect *r;
struct vwsurf *surface;
struct tms start;
time_t btime;
struct tms stop;
time_t stime;
/*    I N I T I A L I Z E    G R A P H I C S    */
int gx,gy;
gopen()
  {
  struct vwsurf *get_surface();
  surface = get_surface();
  if(       initialize_core(BASIC,NOINPUT,TWOD))  exit(1);
  if(initialize_view_surface(surface,FALSE))   exit(2);
  if(    select_view_surface(surface,FALSE))  exit(3);
  set_viewport_2(0.01,0.99,0.01,0.74);
  set_window(0.,1023.,0.,1023.);
  create_temporary_segment();
  set_rasterop(NORMAL);
  gx = 1024;
  gy = 768;
  }
/* C L O S E  G R A P H I C S and E N D  S E S S I O N  */
gclose()
  {
  close_temporary_segment();
  sleep(2);
  deselect_view_surface(surface);
  terminate_view_surface(surface);
  terminate_core();
  }
gsize(x,y)
  int *x,*y;
  {
  *x=gx;
  *y=gy;
  }
gerase()
  {
  new_frame();
  }
gline(lastx,lasty,x,y)
int lastx,lasty,x,y;
  {
  move_abs_2((double)lastx,(double)lasty);
  line_abs_2((double)x    ,(double)y    );
  }
gmove(lastx,lasty)
int lastx,lasty;
  {
  move_abs_2((double)lastx,(double)lasty);
  }
gdraw(x,y)
int x,y;
  {
  line_abs_2((double)x    ,(double)y    );
  }
struct vwsurf *get_surface()
  {
  if(getenv("WINDOW_ME"))
      return(&windowsurface);
  else
      return(&rawsurface);
  }
glock()
  {
  }
gunlock()
  {
  }
/*  Function to start an interval timer in 1/60 sec ticks
gstart_timer()
  {
  btime=times(&start);
  btime=start.tms_utime  +  start.tms_stime
       +start.tms_cutime +  start.tms_cstime;
  }
/*  Function to stop an interval timer in 1/60 sec ticks
/*         return  number of ticks as integer
gstop_timer()
  {
  int ticks;
  stime=times(&stop);
  stime=stop.tms_utime  +  stop.tms_stime
       +stop.tms_cutime +  stop.tms_cstime;
  ticks=stime-btime;
  return(ticks);
  }
```

## Appendix B - Tables of Results

### Table 1. Systems

| System Description | Proc. | FP/vec | MHz | RAM MB | Disk MB | O/S | Graphics | LAN | Notes |
|---|---|---|---|---|---|---|---|---|---|
| Cray XMP 48 | Cray | vec | 111 | 64 | 160000 | LTSS | TMDS | NSC | 1 |
| Cray 1S | Cray | vec | 83 | 8 | 10000 | LTSS | TMDS | NSC | 1 |
| VAX 11/780 | VAX | FPA | | 4 | 456 | VMS | | Ethernet | |
| Apollo dn 660 | Apollo | none | | 2 | 154 | Domain | Apollo | Apollo Ring | |
| Tektronix 6130 | 32016 | 32081 | 10 | 3 | 80 | Unix 4.2 | Tektronix | Ethernet | |
| Apollo dn 320 | 68010 | PEB | 10 | 1.5 | 70 | Domain | Apollo | Apollo Ring | 2 |
| Apollo dn 550 | 68010 | PEB | 10 | 3 | remote | Domain | Apollo | Apollo Ring | 2 |
| SUN 2+ | 68010 | sky | 10 | 2 | 70 | Unix 4.2 | Sun | Ethernet | |
| IBM PC/AT | 80286 | 80287 | 6 | .64 | 33 | DOS | PGB | Ethernet | 3 |
| SUN 2 | 68010 | none | 10 | 2 | remote | Unix 4.2 | Sun | Ethernet | |
| IBM PC | 8086 | none | 4.77 | .64 | 10 | DOS | C/GDA | Ethernet | 4 |

### Table 2. Processing Speeds

| | LLNL Fortran Kernels in KFLOPS | | | | LINPACK | | Benchmark 3 | | |
|---|---|---|---|---|---|---|---|---|---|
| System Description | Low | High | Ave | Harm | KFLOPS | SASI | KTPS | KFLOPS | Notes |
| Cray XMP 48 | 2615 | 162193 | 47603 | 9283 | 21000 | 120 | 452 | 14905 | |
| Cray 1S | 2283 | 95294 | 28437 | 7485 | 12000 | 88 | 361 | 11913 | |
| VAX 11/780 | 42 | 359 | 148 | 112 | 130 | 1.0 | | | |
| Apollo dn 660 | 45 | 225 | 116 | 102 | 69 | 0.6 | 1.5 | 49 | |
| Tektronix 6130 | 28 | 91 | 49 | 45 | | | | | |
| Apollo dn 320 | 12 | 81 | 44 | 36 | 28 | 0.3 | 0.76 | 25 | |
| Apollo dn 550 | 11 | 76 | 40 | 32 | 47 | 0.3 | | | |
| SUN 2+ | 12 | 53 | 28 | 25 | 22 | | 0.74 | 24 | |
| IBM PC/AT | 3 | 22 | 14 | 12 | 9 | | 0.67 | 22 | |
| SUN 2 | 7 | 22 | 12 | 11 | 6 | | 0.24 | 7.8 | |
| IBM PC | 0.8 | 4 | 2 | 2 | 6 | | | | |

### Table 3. Benchmark 1

| | Lowest Level Graphics | | Library Graphics | | Window Graphics | | |
|---|---|---|---|---|---|---|---|
| System Description | VPS L | VPS S | VPS L | VPS S | VPS L | VPS S | Notes |
| Cray XMP 48 | 2334 | 27321 | 4600 | 18602 | | | |
| Cray 1S | 1916 | 22854 | 3468 | 14735 | | | |
| VAX 11/780 | | | | | | | |
| Apollo dn 660 | | | | | | | |
| Tektronix 6130 | | | | | | | |
| Apollo dn 320 | | | | | | | |
| Apollo dn 550 | | | | | | | |
| SUN 2+ | 530 | 2047 | 161 | 207 | 100 | 216 | |
| IBM PC/AT | | | | | | | |
| SUN 2 | 218 | 830 | 141 | 245 | 125 | 149 | |
| IBM PC | | | | | | | |

Table 4. Benchmark 2, The Sierpinski Curves

| System Description | Algorithm VPS | Lowest Level VPS | Library Graphics VPS | Window Graphics VPS | Notes |
|---|---|---|---|---|---|
| Cray XMP 48 | 174485 | 59122 | 20187 | | |
| Cray 1S | 132616 | 46103 | 16039 | | |
| VAX 11/780 | | | | | |
| Apollo dn 660 | 23960 | 11900 | | 4160 | |
| Tektronix 6130 | | | | | |
| Apollo dn 320 | 8010 | 2890 | | 1633 | |
| Apollo dn 550 | 6390 | 3800 | | 1580 | |
| SUN 2+ | 16512 | 3002 | 238 | 154 | |
| IBM PC/AT | 15612 | 1542 | 481 | | |
| SUN 2 | 15285 | 1015 | 250 | 177 | |
| IBM PC | 5587 | 537 | 145 | | |

Table 5. The Space Shuttle, standard plot.

| System Description | Lowest Level Graphics 2D KPS | Lowest Level Graphics 3D KPS | Library Graphics 2D KPS | Library Graphics 3D KPS | Window Graphics 2D KPS | Window Graphics 3D KPS | Notes |
|---|---|---|---|---|---|---|---|
| Cray XMP 48 | 670 | 561 | 422 | 379 | | | |
| Cray 1S | 550 | 464 | 342 | 307 | | | |
| VAX 11/780 | | | | | | | |
| Apollo dn 660 | | | | | | | |
| Tektronix 6130 | | | | | | | |
| Apollo dn 320 | | | | | | | |
| Apollo dn 550 | | | | | | | |
| SUN 2+ | 10.8 | 6.9 | 8.0 | 5.7 | 5.6 | 4.5 | |
| IBM PC/AT | | | | | | | |
| SUN 2 | 6.2 | 2.8 | 5.9 | 2.7 | 6.0 | 2.1 | |
| IBM PC | | | | | | | |

## Notes

1. The operating system is the Livermore TimeSharing System (LTSS) and the graphics is the Television Monitor Display System (TMDS) with 256 channels of 512x512x1 bit frame buffer. The LAN is the hyper-channel from NSC.

2. The PEB is the Performance Enhancement Board.

3. Professional Graphics Board (PGB) with the Halo graphics library.

4. Color/Graphics Display Adapter (C/GDA) with the Halo graphics library.